JOAuth2授权加密办法

护理平台接入指南

目录

[下载 3](#_Toc4683012)

[包含的内容 4](#_Toc4683013)

[快速上手 4](#_Toc4683014)

# 下载

JOAuth2授权平台提供jar包帮开发者快速上手并进行二次开发，因此任何想要集成JOAuth2授权的项目应首先下载以下jar包：

* UserAgentUtils-1.21.jar
* hutool-all-4.5.0.jar
* joauth2-client.1.0.4.jar(以最新版本为准)

除joauth2-client.jar外，所有的jar包版本以本文档为准，不得随意更改，错误的版本可能导致意想不到的兼容性问题。

对于maven项目，在pom.xml中添加如下配置：

<dependency>  
 <groupId>com.github.njedu</groupId>  
 <artifactId>joauth2-client</artifactId>  
 <version>1.0.4</version>  
</dependency>

<repositories>  
 <repository>  
 <id>jitpack.io</id>  
 <url>https://jitpack.io</url>  
 </repository>  
</repositories>

maven项目只需要手动引入joauth2-client，其他依赖会自动引入，joauth2-client的版本以最新版本为准。

# 包含的内容

下载的joauth2-client-\*.jar，是授权调用方法主要的jar包，目录结构如下：

com

├──joauth2

│ ├──Client.class

│ ├──ClientUser.class

│ ├──ClientLogin.class

│ ├──ClientDog.class

│ ├──AuthSecureUtil.class

│ ├──JOAuthListener.class

│ ├──OAuth2Constants.class

│ ├──R.class

│ └──UserAgentGetter.class

└──softkey

├──juyunew3.class

└──DesUtils.class

hutool.jar和UserAgertUtils.jar为授权依赖的第三方jar包，不提供目录结构。

# 快速上手

1. 在classes根路径创建application.properties，包含如下内容：

#加密狗

dog.beginpwd=B30FA70B10A8AAC6080BE9FE67749A0A

dog.endpwd=C328CE80CBB36F9AC1935826325D7D2F

dog.lockid=ECD27F9367E5B31FDB7E6AF82DE3C3FA

#WEB授权

auth.app\_key=14d37d91824643cca6e8cc54aea7493d

auth.app\_secret=172b86f7577c1c393ad3d521df2957f0

auth.url=http://www.njedusoft.com/joauth2/connect/oauth2

auth.redirect\_uri=http://127.0.0.1:8080/vcare/connect/auth/getToken

auth.app\_encrypt=7ebc38a51beac302ba018c5d1e693bcf57210f8295d6fcf8a0dbbc544be7b4173431e41379f9b7ad9c991f5863bb6852d02b020752a097dbce3380b6792aa3964638bf65367a4bbe78ddbbea2c59cb38

其中dog.\* 是加密狗的相关配置，auth.\* 是WEB授权的相关配置，具体解释如下：

* bog.beginpwd：加密狗读密码前半部分（加密狗的字符串）
* dog.endpwd：加密狗读密码后半部分（加密后的字符串）
* dog.lockid：MD5加密后的加密狗锁ID（参见操作指南PPT）
* auth.app\_key：创建应用后提示的app key
* auth.app\_secret: 创建应用后提示的app secret
* auth.redirect\_uri: 固定格式，http://【应用IP】:【应用端口】/connect/auth/getToken
* auth.url：http://www.njedusoft.com/joauth2/connect/oauth2
* auth.app\_encrypt：（加密后数据）创建应用后提示的app encrypt

1. 修改登录方法，使用自身的业务判断账号密码是否正确，如果正确，调用如下方法：

@RequestMapping("login/in")  
@ResponseBody  
public ResultBean doLogin(String username, String password){  
 User user = service.login(username, password);  
 if (user != null){  
 // 调用授权登录逻辑  
 ClientUser<User> clientUser = new ClientUser<User>();

// 备注信息  
 clientUser.setExtMsg("用户登录");  
 clientUser.setId(user.getId());  
 clientUser.setNickname(user.getNickname());  
 clientUser.setUsername(username);

// session attribute key  
 clientUser.setSessionAttrName(“user”);

// 登录的用户对象  
 clientUser.setUser(user);  
 R r = ClientLogin.login(clientUser, request);

// 如果code=200，代表成功，其他都是错误  
 if (r.getCode() == 200) {  
 // 成功  
 return ResultBean.ok(null);  
 } else {  
 // 失败，进行提示  
 return new ResultBean(101, r.getMsg());  
 }  
 }  
 return new ResultBean(101, "用户名或密码错误");  
}

1. 修改注销方法：

@RequestMapping("logout")

public ModelAndView doLogout(ModelAndView modelAndView){

// 退出账号

User user = (User)session.getAttribute(User.CURRENT\_LOGIN\_USER);

// 调用ClientLogin.logout()方法

ClientLogin.logout(user.getId(), session, User.CURRENT\_LOGIN\_USER);

modelAndView.setViewName("admin/login");

return modelAndView;

}

1. 修改过滤器（拦截器），对于没有授权许可的请求禁止通行：

// 检查授权许可

if (Client.MAX\_USER == 0) {

session.removeAttribute(“user”);

response.sendRedirect(base + "/admin/login");

return false;

}

1. 修改登录页面，添加相应提示